Department of Computing and Software

Faculty of Engineering --- McMaster University

A Polynomial Time Jacobi Method for Lattice Basis Reduction

by

Zhaofei Tian, Wen Zhang, and Sanzheng Qiao

CAS Report Series **CAS-12-04-SQ**

Department of Computing and Software **Sept. 2012**
Information Technology Building
McMaster University
1280 Main Street West, Hamilton, Ontario, Canada L8S 4K1

A Polynomial time Jacobi Method for Lattice Basis Reduction

Zhaofei Tian
Department of Computing and Software, McMaster University
Hamilton, Ontario, L8S 4K1, Canada
Wen Zhang
Institute of Mathematics, School of Mathematical Science
Fudan University, Shanghai, 200433, P.R.China
Sanzheng Qiao
Department of Computing and Software, McMaster University
Hamilton, Ontario, L8S 4K1, Canada

Abstract

Among all lattice reduction algorithms, the LLL algorithm is the first and perhaps the most famous polynomial time algorithm, and it is widely used in many applications. In 2012, S. Qiao [24] introduced another algorithm, the Jacobi method, for lattice basis reduction. S. Qiao and Z. Tian [25] improved the Jacobi method further to be polynomial time but only produces a Quasi-Reduced basis. In this paper, we present a polynomial time Jacobi method for lattice basis reduction (short as Poly-Jacobi method) that can produce a reduced basis. Our experimental results indicate that the bases produced by Poly-Jacobi method have almost equally good orthogonality defect as the bases produced by the Jacobi method.

1 Introduction

Recently, mathematical methods connecting different research fields are more and more popular. A typical example is the lattice based method, which connects lattices with many other fields, and hence involved in many applications, such as Cryptography [10, 3], Global Positioning System (GPS) [29, 8] and Wireless Communications [28]. A *lattice* is an infinite set of discrete points (or vectors from the origin) in Euclidean Space. Those points can be represented by integer linear combinations of a finite set of linearly independent vectors [17] called a *basis*. We define the number of vectors in a basis the *dimension* of the lattice, which is fixed for a lattice. A lattice of dimension at least 2 has infinitely many bases [10]. We say a lattice vector is shorter than another vector, if the *Euclidean length* of this vector is less than the length of the other vector. A non-zero vector is called a *shortest vector* if its length is the shortest among the lengths of all other lattice vectors. The goal of *lattice basis reduction algorithms* is to find a basis with more orthogonal (or shorter) vectors, depending on different measurements.

In 1850's, in his second letter to Jacobi, Hermite gave the first formal definition of a reduced basis for a lattice. Korkine and Zolotareff [15] strengthened the definition based on the Hermite's original definition in 1870's, which is now called the *HKZ reduced basis*. In 1890's, Minkowski introduced a reduced basis

[19, 5, 14], known as the *Minkowski reduced basis* later. It requires the shortest vectors that can form a basis for the lattice. Thus, the definition of the Minkowski reduced basis is the strongest one among all the definitions.

The problem is NP-complete [1] to find a Minkowski reduced basis for the lattices of dimension at least 3. Even to find a weaker HKZ reduced basis needs exponential time [1]. In 1983, R. Kannan presented an algorithm that computes a HKZ reduced basis [13], which used lattice bases projection strategy. B. Helfrich [9] in 1985 and R. Kannan in 1987 [14] improved this algorithm to computes both Minkowski reduced bases and HKZ reduced bases, respectively. Practical algorithms for computing Minkowski reduced and HKZ reduced lattice bases can be found in [27], which use sphere decoding technique to find a shortest vector in a lattice.

All of the above defined reduced bases need exponential time to be computed. Another category of the reduced bases is the bases that can be computed in polynomial time. The LLL reduced basis is a typical reduced basis, named after the three authors A. Lenstra, W. Lenstra, and L. Lovász, which can be found by a polynomial time algorithm presented in 1982 [16]. It is commonly believed that the LLL algorithm is the first polynomial time algorithm for lattice basis reduction. Theoretically, it can produce only an approximate shortest vector which is about a factor of at most $\mathcal{O}(2^n)$ [2] times longer than a shortest vector of the lattice. However, it performs well in practice. Hence the LLL algorithm becomes an important tool in many applications, such as cryptography [21], sphere decoding [23], integer programming [22], and so on. There are also many other algorithms based on the LLL algorithm, improving the original LLL algorithm in either efficiency or accuracy.

In 2012, S. Qiao [24] presented a *Jacobi method for lattice basis reduction* (short as *Jacobi method* in the following sections) to produce a reduced basis, which adopts a different strategy from the LLL algorithm. It chooses the *Lagrange reduction algorithm* [20] as a reduction tool for every two vectors in a basis and produces a reasonably good basis. A *Quasi-Jacobi method* [25] for lattice basis reduction based on the Jacobi method is introduced further, which is proved to be a polynomial time algorithm, but only produces a weaker Quasi-Reduced basis [25]. In this paper, we present a polynomial time Jacobi algorithm (short as the *Poly-Jacobi method*) that produces a reduced basis defined in [24]. It is shown in our experimental results that the two algorithms, the Poly-Jacobi method presented in this paper and the Jacobi method introduced by S. Qiao [24], produce almost equally good bases according to the measurement criteria.

The paper is organized as follows. In this section, we introduce some basic concepts including lattice, bases and lattice reduction algorithms. In the next section, the Lagrange reduction algorithm and the Jacobi reduction algorithm are presented. The polynomial time Jacobi method is given in section 3. In section 4, we compare the Poly-Jacobi method with the Jacobi method in terms of orthogonality defect (or Hadamard Ratio) and running time. Finally, the paper is concluded in section 5.

2 Jacobi Methods for Lattice Basis Reduction

We recall in detail the Lagrange reduction algorithm [20] for two dimensional lattices and the Jacobi reduction algorithm [24] in this section. The two algorithms use a matrix transformation technique for vector operations called the unimodular transformation [27], which is also recalled in the first part of this

section. The unimodular transformation is also used in the Poly-Jacobi method introduced in the next section.

2.1 Unimodular Transformation

We use the matrix form of a basis called a basis matrix, or a lattice generator matrix in this paper. That is, we construct a matrix by forming its columns with the vectors in a basis. Let A be a basis matrix for a given lattice L. If the dimension of the lattice L is at least 2, we know there are more than one bases for L. A nice property for the bases is that the *volume*s of the parallelepipeds they generate are equal [17, 10]. We call this volume the *determinant* of the lattice, denoted by $\det(L)$, or $|\det(A)|$ in terms of the basis matrix.

Let B be another basis matrix for the lattice L, then $|\det(A)| = |\det(B)|$. Since B is a basis matrix for L, the columns of A can be represented by the integer linear combinations of columns in B. Thus we can always find an integer coefficient matrix Z, called a *unimodular* matrix, such that $B = AZ$. Similarly, there exists another integer coefficient matrix $\bar{Z}$ such that $A = B\bar{Z}$. Obviously, $\bar{Z}$ is the inverse of the matrix Z. Since $|\det(A)| = |\det(B)|$, we have $\det(Z) = \pm 1$.

Definition 2.1 (Unimodular)**.** A nonsigular integer matrix M is called *Unimodular* if $\det(M) = \pm 1$.

Having this definition, the purpose of lattice reduction algorithms can be regarded as to find a unimodular matrix that transforms the given basis into a more orthogonal one.

Since there are more than one basis for a lattice of dimension greater than or equal to 2, it is necessary to define a measurement of the orthogonality of a basis. In this paper, the measurement we adopt is called the *Hadamard Ratio* [10] or the *orthogonality defect*.

Definition 2.2 (Hadamard Ratio [10])**.** Given a basis matrix $A = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$ for a lattice L, the *Hadamard Ratio* of the basis matrix A is defined as

$$\mathscr{H}(A) = \left(\frac{\det(L)}{||\mathbf{a}_1||_2 \cdot ||\mathbf{a}_2||_2 \cdots ||\mathbf{a}_n||_2}\right)^{\frac{1}{n}} . \tag{2.1}$$

According to the *Hadamard's Inequality* [11], we always have $\det(L) \leq ||\mathbf{a}_1||_2 \cdot ||\mathbf{a}_2||_2 \cdots ||\mathbf{a}_n||_2$. Hence we get $0 < \mathscr{H}(A) \leq 1$. The above equality holds if and only if $\mathbf{a}_i$ $(1 \leq i \leq n)$ are orthogonal to each other. Geometrically, the Hadamard Ratio measures the scaled geometric mean of the lengths of columns in A. Therefore, the shorter the mean of the columns is, the closer the Hadamard Ratio (2.1) is to 1, and the more orthogonal the columns in A are.

2.2 Lagrange Reduction Algorithm

Given two integers, the *Euclidean algorithm* [26] can compute their the greatest common divisor in polynomial time. The similar idea is borrowed in the Lagrange lattice reduction algorithm [24]. For a given two dimensional lattice, the algorithm computes an optimally (Minkowski) reduced basis in polynomial time [6, 20].

Definition 2.3 (Lagrange Reduced Basis [18, 24])**.** Given a two-dimensional lattice L and its basis matrix $A = [\mathbf{a}_1,\ \mathbf{a}_2]$, A is called *Lagrange reduced* (or *L-Reduced*), if it satisfies

$$||\mathbf{a}_1||_2 \le ||\mathbf{a}_2||_2 \quad \text{and} \quad |\mathbf{a}_1^T\mathbf{a}_2| \le \frac{1}{2}||\mathbf{a}_1||_2^2, \tag{2.2}$$

where $\mathbf{a}_1^T\mathbf{a}_2$ represents the *inner product* of $\mathbf{a}_1$ and $\mathbf{a}_2$.

For two dimensional lattices, a Lagrange reduced basis is also a Minkowski reduced basis [21]. If we denote θ the angle between $\mathbf{a}_1$ and $\mathbf{a}_2$, we have $|\cos(\theta)| = |\mathbf{a}_1^T\mathbf{a}_2|/(||\mathbf{a}_1||_2 \cdot ||\mathbf{a}_2||_2) \le |\mathbf{a}_1^T\mathbf{a}_2|/||\mathbf{a}_1||_2^2 \le 1/2$. Thus, the angle between the two vectors in a Lagrange reduced basis is located in the range $[\pi/3, 2\pi/3]$ [24].

Let $A = [\mathbf{a}_1,\ \mathbf{a}_2]$ be a basis matrix for a given a lattice L, the following Lagrange reduction algorithm produces a Lagrange reduced basis matrix $[\mathbf{a}_1, \mathbf{a}_2]$ for the lattice L, which overwrites the original input matrix $[\mathbf{a}_1, \mathbf{a}_2]$.

Algorithm 1: Lagrange reduction algorithm for two dimensional lattices

Input : A lattice basis matrix $[\mathbf{a}_1, \mathbf{a}_2]$
Output: A Lagrange reduced basis matrix $[\mathbf{a}_1, \mathbf{a}_2]$

1 **if** $||\mathbf{a}_1||_2 < ||\mathbf{a}_2||_2$ **then**
2 **SWAP**$(\mathbf{a}_1,\ \mathbf{a}_2)$;
3 **repeat**
4 Set $q = \lfloor \mathbf{a}_1^T\mathbf{a}_2/||\mathbf{a}_2||_2^2 \rceil$;
5 $\mathbf{Z}_{12} = \begin{bmatrix} 0 & 1 \\ 1 & -q \end{bmatrix}$;
6 $[\mathbf{a}_1, \mathbf{a}_2] \leftarrow [\mathbf{a}_1, \mathbf{a}_2]\mathbf{Z}_{12}$;
7 **until** $||\mathbf{a}_1||_2 \le ||\mathbf{a}_2||_2$;

When the algorithm terminates, the given basis matrix $[\mathbf{a}_1, \mathbf{a}_2]$ above is overwritten by a Lagrange reduced basis matrix in line 6. The notation $\lfloor \cdot \rceil$ in line 4 represents the nearest integer rounding. The algorithm keeps reducing the length of the longer vector $\mathbf{a}_1$ and swapping the two vectors by computing $[\mathbf{a}_1, \mathbf{a}_2]\mathbf{Z}_{12}$, until it cannot be reduced by introducing an integer scalar q, i.e., $||\mathbf{a}_1||_2 \le ||\mathbf{a}_2||_2$. In each iteration from line 4 to line 6, the length of $\mathbf{a}_1$ is reduced with a factor of at least $\sqrt{3}$, except the first and the last iterations [25]. The uncertainty of those two iterations make the complexity analysis of the Jacobi method to be difficult.

In fact, the Lagrange algorithm is a greedy algorithm [20]. It finds an integer scalar q that minimizes $||\mathbf{a}_1 - q\mathbf{a}_2||_2$ in each iteration. Thus, a global optimum is produced when the algorithm terminates, which is also a Minkowski reduced basis.

2.3 Jacobi Method for Lattice Basis Reduction

In 1846, C. Jacobi originally proposed a method for solving eigenvalue problems of real symmetric matrices [12, 7], which is called the *Jacobi method* later. The Jacobi method iteratively performs a symmetric

eigenvalue decomposition on every 2×2 submatrix until the symmetric matrix becomes almost diagonal. In 2012, S. Qiao [24] introduces a Jacobi method for lattice basis reduction, which is similar as the process that the original Jacobi method does for solving the eigenvalue problem. Given a lattice basis, it invokes the Lagrange algorithm to reduce every pair of vectors, and hence produces a reduced basis. Each pair of vectors in the new basis is Lagrange reduced.

For a given general n dimensional lattice, the Jacobi method produces a reduced basis defined as follows.

Definition 2.4 ([24])**.** Given an n dimensional lattice L and its basis matrix $A = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$, A is called reduced if

$$||\mathbf{a}_i||_2 \leq ||\mathbf{a}_j||_2, \tag{2.3}$$

$$|\mathbf{a}_i^T \mathbf{a}_j| \leq \frac{1}{2}||\mathbf{a}_i||_2^2, \tag{2.4}$$

for all $1 \leq i < j \leq n$.

The condition (2.3) in the above definition implies that the vectors of the reduced basis are sorted by their Euclidean lengths. Since the condition (2.4) focuses on every two vectors locally, we firstly present a modified Lagrange reduction algorithm [24] as a procedure to reduce each pair of vectors. For a given lattice basis matrix $A = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$, we construct a *Gram Matrix* $G = A^T A$. Denote $[g_{ij}]$ the elements in the Gram matrix, they have some special properties, such as $g_{ij} = g_{ji} = \mathbf{a}_i^T \mathbf{a}_j$ and $g_{ii} = ||\mathbf{a}_i||_2^2$. The Lagrange procedure will be called on every 2×2 submatrix of G.

The following procedure runs the Lagrange reduction algorithm on a given pair of vectors of indices (i, j).

Algorithm 2: Lagrange2(G, i, j)

Input : A *Gram* matrix $G = A^T A$ and two indices i, j
Output: A unimodular matrix Z_{ij} such that the pair of ith, jth vectors in AZ_{ij} is L-reduced

$Z_{ij} = I_n$;
if $g_{ii} < g_{jj}$ **then**
    Swap $G(:, i)$ and $G(:, j)$;
    Swap $G(i, :)$ and $G(j, :)$;
    Swap $Z(:, i)$ and $Z(:, j)$;
repeat
    Set $q = \lfloor g_{ij}/g_{jj} \rceil$;
    Set Z to be I_n except $z_{ii} = 0, z_{jj} = -q$ and $z_{ij} = z_{ji} = 1$;
    $G \leftarrow Z^T G Z$;
    $Z_{ij} \leftarrow Z_{ij} Z$;
until $g_{ii} \leq g_{jj}$;

Using the Algorithm 2 as the reduction procedure in 2×2 submatrices of the Gram matrix G, the following Algorithm 3 introduced by S. Qiao [24] computes a reduced basis defined in the Definition 2.4.

Algorithm 3: Jacobi reduction algorithm

Input : A basis matrix $A = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$

Output: A unimodular matrix $\mathbf{Z}$ such that $A\mathbf{Z}$ forms a Jacobi reduced basis

```
1 Z = I_n ;
2 G = A^T A ;
3 while not all pairs (a_i, a_j) satisfy the Jacobi reduced condition (2.4) do
4     for i ← 1 to n − 1 do
5         for j ← i + 1 to n do
6             [G, Z_ij] ← Lagrange2(G, i, j) ;
7             Z ← ZZ_ij ;
```

Given a lattice basis matrix A, the Algorithm 3 reads a Gram matrix G and computes a reduced basis we defined. It applies the Lagrange reduction algorithm to all possible pairs of vectors in the basis during the **`while`** loop. It has been showed [24] that the bases it produced are reasonably good comparing with the widely used LLL algorithm.

The convergence of the Jacobi reduction algorithm is proved in [25]. However, due to the unconcerned reduction factor in the last iteration of the Lagrange algorithm, the Jacobi method cannot be proven to be a polynomial time algorithm. For example, we are given a basis $A = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3]$. For the first round of the **`while`** loop, the execution of the Jacobi method might invoke the Lagrange reduction algorithm to reduce every pair of vectors in the order $(\mathbf{a}_1, \mathbf{a}_2), (\mathbf{a}_1, \mathbf{a}_3), (\mathbf{a}_2, \mathbf{a}_3)$. If the calls of **`Lagrange2(G,i,j)`** in line 6 only involve the first and last iterations, we cannot guarantee a fixed factor for the reduction of any two vectors, even though their lengths are reduced. Hence the call of $(\mathbf{a}_1, \mathbf{a}_2), (\mathbf{a}_1, \mathbf{a}_3), (\mathbf{a}_2, \mathbf{a}_3)$ may repeat exponential times. Thus the complexity of the Jacobi method may not be polynomial.

3 A Polynomial Time Jacobi Algorithm to Produce a Reduced Basis

The paper [25] also introduces a weaker Jacobi reduction algorithm called the *Quasi-Jacobi reduction algorithm* (short as the *Quasi-Jacobi method*). However, even though the Quasi-Jacobi method is polynomial time, it can only produce a quasi-reduced basis. In this section, we present a polynomial time algorithm that produces a reduced basis.

3.1 Quasi-Lagrange Reduction Algorithm

A weaker Lagrange reduced basis for a two dimensional lattice is defined as the following.

Definition 3.1 (Quasi-Lagrange Reduced Basis [25])**.** A basis matrix $A = [\mathbf{a}_1, \mathbf{a}_2]$ for a two dimensional lattice L is called *Quasi-Lagrange Reduced* (or *QL-Reduced*), if it satisfies

$$w||\mathbf{a}_1||_2 \leq ||\mathbf{a}_2||_2, \tag{3.1}$$

$$|\mathbf{a}_1^T \mathbf{a}_2| \leq \frac{1}{2}||\mathbf{a}_1||_2^2 \text{ and } |\mathbf{a}_1^T \mathbf{a}_2| \leq \frac{1}{2}||\mathbf{a}_2||_2^2, \tag{3.2}$$

where $\frac{1}{\sqrt{3}} \leq w < 1$.

Comparing with the definition of the Lagrange reduced basis, the Quasi-Lagrange reduced basis adds a parameter w, which makes it weaker because in a Lagrange reduced basis the first vector will always strictly less or equal than the second vector. Since the domain of the parameter w is defined as $\frac{1}{\sqrt{3}} \leq w < 1$, so a Lagrange reduced basis must be a Quasi-Lagrange reduced basis. It has been proven that each iteration of the Lagrange reduction algorithm can reduce the length of one of the two vectors with a factor of at least $\sqrt{3}$, except the first and the last iterations [25]. The lower bound $\frac{1}{\sqrt{3}}$ is chosen for w to guarantee that the last iteration will reduce the vectors with a factor of at least w, and hence a polynomial time algorithm, the Poly-Jacobi method, can be designed.

Let $A = [\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n}]$ to be a lattice generator matrix, and let $G = A^T A$ to be a Gram matrix. Based on the Lagrange reduction algorithm, the following Algorithm 4 constructs a pair of Quasi-Lagrange reduced vectors on the given indices (i, j) for the two dimensional sublattice of the lattice generated by A in polynomial time.

Algorithm 4: Quasi-Lagrange2(G, i, j, w)

Input : A *Gram* matrix $G = A^T A$, two indices i, j and w
Output: A unimodular matrix Z_{ij} such that the pair of ith, jth vectors in AZ_{ij} is QL-reduced

1 $Z_{ij} = I_n$;
2 **if** $g_{ii} < g_{jj}$ **then**
3 **Swap** $G(:, i)$ and $G(:, j)$;
4 **Swap** $G(i, :)$ and $G(j, :)$;
5 **Swap** $Z(:, i)$ and $Z(:, j)$;
6 **repeat**
7 Set $q = \lfloor g_{ij}/g_{jj} \rceil$;
8 **if** $|q| \leq 1 \wedge w^2 g_{ii} \leq g_{jj}$ **then**
9 Break the **repeat** loop;
10 Set Z to be I_n except $z_{ii} = 0, z_{jj} = -q$ and $z_{ij} = z_{ji} = 1$;
11 $G \leftarrow Z^T G Z$;
12 $Z_{ij} \leftarrow Z_{ij} Z$;
13 **until** $g_{ii} \leq g_{jj}$;

It is worth to mention that the new Quasi-Lagrange Algorithm 4 adds one more terminating condition inside the **`while`** loop on line 12 and 13. The convergence proof [25] of the Lagrange reduction algorithm tells us that the integer scalar $|q| \leq 1$ indicates the last iteration of the algorithm. Hence the condition $|q| \leq 1 \wedge w^2 g_{ii} \leq g_{jj}$ can reduce the lengths of vectors maximumly.

3.2 A Polynomial Time Jacobi Method

Using the Algorithm 4 as a reduction tool in every 2×2 submatrix, the following Algorithm 5, the Poly-Jacobi method, computes a reduced basis defined in Definition 2.4.

Algorithm 5: A polytime algorithm to produce a reduced basis

Input : A basis matrix $A = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]$ and w

Output: A unimodular matrix $\mathbf{Z}$ such that $A\mathbf{Z}$ forms a reduced basis

1 $\mathbf{Z} = I_n$;
2 $G = A^T A$;
3 **while** *not all pairs* $(\mathbf{a}_i, \mathbf{a}_j)$ *satisfy the Quasi-Lagrange reduced condition* (3.2) **do**
4 **for** $i \leftarrow 1$ **to** $n-1$ **do**
5 **for** $j \leftarrow i+1$ **to** n **do**
6 $[\mathbf{G}, \mathbf{Z}_{ij}] \leftarrow$ Quasi-Lagrange2($\mathbf{G}, i, j, w$) ;
7 $\mathbf{Z} \leftarrow \mathbf{Z}\mathbf{Z}_{ij}$;
8 Set $G \leftarrow \mathbf{Z}_s^T G \mathbf{Z}_s$ such that the diagonal of G is sorted ;
9 $\mathbf{Z} \leftarrow \mathbf{Z}\mathbf{Z}_s$

There are two differences between the Jacobi reduction algorithm 3 [25] and the above Poly-Jacobi reduction algorithm 5. Firstly, in line 6, the Jacobi method invokes the Lagrange reduction algorithm, but the Poly-Jacobi method invokes the Quasi-Lagrange reduction algorithm, which makes the Poly-Jacobi method to be polynomial, as shown in the section 3.3. Secondly, the Poly-Jacobi method sorts the vectors by their lengths in line 8 and 9.

To show the basis produced by the Algorithm 5 is reduced, we can check the two conditions in the definition 2.4. The condition (2.3) requires all vectors in a reduced basis should be sorted, which is satisfied by the line 8 and 9. The other condition (2.4) is similar with the condition (3.2) of a Quasi-Lagrange reduced basis 2.4. Hence it is satisfied as well. Therefore, in the end of the Algorithm 5, a reduced basis is generated.

3.3 Complexity Analysis

To show the Poly-Jacobi method is polynomial time, we analyze its complexity globally including the complexity of a single iterative step in the Quasi-Lagrange reduction algorithm, the maximum rounds of the **`while`** loop in the Poly-Jacobi method and the complexity of sorting vectors.

1. Complexity of one single step Quasi-Lagrange iteration

   The convergence proof in [25] tells us that the total cost of one single step of the Quasi-Lagrange iteration is $\mathcal{O}(n)$.

2. **`while`** loop analysis of the Poly-Jacobi method

   By the Definition 3.1, we have $1/w^2 \leq 3$. So each iteration in the Quasi-Lagrange algorithm will reduce the length of one of the two vectors with the factor at least $\sqrt{3}$. In each round of the **`while`** loop, the procedure **`Quasi-Lagrange2(G,i,j,w)`** will be invoked at most $\mathcal{O}(n^2)$ times.

   Denote $B = \max_{1 \leq i \leq n}(g_{ii})$ and $D = \Pi_{i=1}^{n}(g_{ii})$. Then each iteration of the Quasi-Lagrange algorithm reduces D with a factor of at least $\min(1/w^2, 3)$. Hence, the maximum rounds of the **`while`** loop is

$\mathcal{O}(nlogB)$. Therefore, the **while** loop contributes a complexity factor of $\mathcal{O}(n^3 logB)$ in the worst case.

3. Sorting vectors
   A typical sorting algorithm requires $\mathcal{O}(n \log n)$ arithmetic operations [4].

To sum up, the complexity of the Poly-Jacobi reduction algorithm is $\mathcal{O}(n^4 logB)$, which proves it is a polynomial time algorithm.

4 Experimental Results

Comparing with the Jacobi method, the Poly-Jacobi method lacks of the last iteration when it invokes the Quasi-Lagrange algorithm. To evaluate the practical efficiency of the Poly-Jacobi method, we compared the Poly-Jacobi method with the Jacobi method. The two methods are implemented in **MATLAB R2010b** running on a **Linux 32-bit** version machine.

The dimension of experimental lattices are chosen as 10, 50, 100, 150 and 200. In each dimension, we called the MATLAB function **rand()** to generate random lattice basis matrices. We run the two algorithms 100 times in each dimension and calculate the average data. The experimental results of the Hadamard Ratio showed that the bases produced by the Poly-Jacobi method produced were almost as good as the ones of the Jacobi method.

	Hadamard Ratio		Runtime	
Dimention	Jacobi	Poly-Jacobi	Jacobi	Poly-Jacobi
10	0.7646	0.6937	0.0004	0.0001
50	0.5002	0.4988	0.0296	0.0071
100	0.4786	0.4787	0.1121	0.0221
150	0.4678	0.4678	0.0989	0.0440
200	0.4626	0.4625	0.3278	0.0716

Table 1: The Hadamard Ratio and runtime of the Jacobi method and the LLL algorithm

The following is a brief discussion of the experimental results.

1. Orthogonality defect

   The experimental data shows the Poly-Jacobi method produces almost identically good bases as the Jacobi reduction algorithm, especially when the dimension of the lattices is large than or equals 50.

2. Running time

   The data of running time also indicates that the Poly-Jacobi method is much fast than the Jacobi method due to the invoking of the Quasi-Lagrange algorithm.

Our experimental data illustrates that the Poly-Jacobi method is significantly faster than the Jacobi method. However, even the Poly-Jacobi lacks the last iteration when it invokes the Quasi-Lagrange algorithm, it produces almost the identically good bases as the Jacobi method.

5 Conclusion

In this paper, we presented a novel polynomial time Jacobi method (short as Poly-Jacobi method) for lattice basis reduction. and evaluated the complexity of the algorithm. The Poly-Jacobi method invokes a weaker Quasi-Lagrange reduction algorithm which is missing the last iteration comparing with the Lagrange reduction algorithm. The complexity analysis shows that the Poly-Jacobi method is polynomial time.

Our experimental results showed that the bases constructed by the Poly-Jacobi method were as good as the bases produced by the Jacobi method in terms of Hadamard Ratio, especially when the dimensions of lattices are greater than or equal 50. The performance compare between the Poly-Jacobi method and the Jacobi method illustrated that the former one is much faster than the other. Future improvement is expected as the investigation for low dimension lattices.

References

[1] Dorit Aharonov and Oded Regev. Lattice problems in NP and co-NP. *J. ACM*, 52:749–765, September 2005.

[2] L. Babai. On Lovasz lattice reduction and the nearest lattice point problem. *Combinatorica*, 6:1–13, 1986.

[3] Dan Boneh, The Rsa Cryptosystem, Invented Ron Rivest, Adi Shamir, Len Adleman, and Was Rst. Twenty years of attacks on the RSA cryptosystem. *Notices of the AMS*, 46:203–213, 1999.

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[5] John L. Donaldson. Minkowski reduction of integral matrices. *j-MATH-COMPUT*, 33(145):201–216, jan 1979.

[6] Steven Galbraith. Mathematics of Public Key Cryptography. An unpublished edited version, April 2012.

[7] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.

[8] Babak Hassibi and Haris Vikalo. On the sphere-decoding algorithm i. expected complexity. *IEEE Trans. Sig. Proc*, pages 2806–2818, 2005.

[9] Bettina Helfrich. Algorithms to construct Minkowski reduced and Hermite reduced lattice bases. *Theoretical Computer Science*, 41:125 – 139, 1985.

[10] J. Hoffstein, J.C. Pipher, and J.H. Silverman. *An introduction to mathematical cryptography*. Undergraduate texts in mathematics. Springer, 2008.

[11] Roger A. Horn and Charles R. Johnson. *Matrix analysis*. Cambridge University Press, New York, NY, USA, 1986.

[12] C.J.G. Jacobi. Über ein leichtes verfahren, die in der theorie der säkularstörungen vorkommenden gleichungen numerisch aufzulösen. *Journal für reine und angewandte Mathematik*, 30:51–95, 1846.

[13] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, STOC '83, pages 193–206, New York, NY, USA, 1983. ACM.

[14] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12:415–440, August 1987.

[15] A. Korkine and G. Zolotareff. Sur les formes quadratiques. *Mathematische Annalen*, 6:366–389, 1873. 10.1007/BF01442795.

[16] A.K. Lenstra, Lenstra, and Lászlo Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

[17] Franklin T. Luk, Sanzheng Qiao, and Wen Zhang. A lattice basis reduction algorithm. Technical report, Institute for Computational Mathematics Hong Kong Baptist University, 2010.

[18] J. Martinet. *Perfect Lattices in Euclidean Spaces*. Springer-Verlag, Berlin, 2003.

[19] H. Minkowski. Discontinuity region for arithmetical equivalence. *J. reine Angew*, (129):220–274, 1905.

[20] P. Q. Nguyen and D. Stehlé. Low-dimensional lattice basis reduction revisited. *ACM Transactions on Algorithms*, 2009. To appear.

[21] Phong Nguyen. Lattice reduction algorithms: Theory and practice. In Kenneth Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 2–6. Springer Berlin / Heidelberg, 2011.

[22] Phong Q. Nguyen and Brigitte Valle. *The LLL Algorithm: Survey and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[23] Sanzheng Qiao. Integer least squares: sphere decoding and the LLL algorithm. In *Proceedings of the 2008 C3S2E conference*, C3S2E '08, pages 23–28, New York, NY, USA, 2008. ACM.

[24] Sanzheng Qiao. A Jacobi method for lattice basis reduction. In *Proceedings of 2012 International Conference on Wireless Communications and Networks*, Xi'an China, May 2012. To appear.

[25] Zhaofei Tian and Sanzheng Qiao. A complexity analysis of a Jacobi method for lattice basis reduction. In *Proceedings of the Fifth International Conference on Computer Science and Software Engineering*, ACM International Conference Proceedings Series, Montreal, Quebec, Canada, June 2012. ACM Press. To appear.

[26] I. M. Vinogradov. *Elements of number theory*. Dover Publications Inc., New York, 1954. Translated by S. Kravetz.

[27] Sanzheng Qiao Wen Zhang and Yimin Wei. Practical algorithms for constructing HKZ and Minkowski reduced bases. Technical report, McMaster University, 2011. CAS-11-04-SQ.

[28] D. Wübben, D. Seethaler, J. Jalden, and G. Matz. Lattice reduction: A survey with applications in wireless communications. *IEEE Signal Processing Magazine*, 28(3):70–91, May 2011.

[29] P. Xu, C. Shi, and J. Liu. Integer estimation methods for GPS ambiguity resolution: an applications oriented review and improvement. *Survey Review*, 44:59–71, Jan. 2012.